UNITED STATES PATENT APPLICATION

FOR

Algorithm for Dynamic Selection of Data Locking Granularity

INVENTORS:

Steve Holmgren

Prepared by:

Blakely, Sokoloff, Taylor & Zafman LLP
32400 Wilshire Boulevard
Seventh Floor
Los Angeles, CA 90025-1026
(408) 720-8300

Attorney's Docket No.: 003399.P089

"Express Mail" mailing label number: EL 867 648 995 US
Date of Deposit: January 25, 2002
I hereby certify that I am causing this paper or fee to be deposited with the United States Postal Service "Express Mail Post Office to Addressee" service on the date indicated above and that this paper or fee has been addressed to the Commissioner for Patents, Washington, D. C. 20231

Carla Zavala
(Typed or printed name of person mailing paper or fee)

(Signature of person mailing paper or fee) 1-25-02

(Date signed)

Algorithm for Dynamic Selection of Data Locking Granularity

FIELD OF THE INVENTION

[0001] The present invention pertains to database technology. More particularly, the present invention relates to improving database concurrency while minimizing a possibility of a deadlock.

BACKGROUND OF THE INVENTION

[0002] Present technology allows multiple users to access one set of data via a network. Capability to access simultaneously large bodies of message data improves the efficiency of this technological development.

[0003] In order to maintain data integrity, no two users may modify data, for example message data in an email system, at the same time. The present technology utilizes locks to serialize data access to one user at a time. To promote the most concurrent access to message data, locks are placed with the finest granularity practical. The difficulty with placing extremely fine grained locks, for example locks on every word of a message data, is the need for single processing entities to obtain and hold multiple locks during the processing of a message. If the granularity of locks within a message system is too fine grained, this leads to deadlock situations.

[0004] A deadlock is a case where one thread of processing holds a lock and, at the same time, requires a lock held by another thread. In addition to holding the lock that the first thread requires, the second thread, in turn, requires the lock that the first thread holds. Without external intervention, this is an

unresolvable situation where the processing of neither thread can progress. In addition, deadlock detection and external intervention slows message processing considerably, thus making the deadlock detection process inefficient.

[0005] A goal of the idealized message processing then is to minimize deadlock conditions, while allowing many threads of processing to access message data or message data infrastructure at the same time, thus improving data concurrency.

SUMMARY OF THE INVENTION

A method and apparatus for improving database concurrency are described. The method may comprise receiving a request to access data, determining a data locality within a database utilizing unique data keys, determining a data locking level based on a deadlock history corresponding to the data locality, and providing access to the data while locking part of the database based on the data locking level.

BRIEF DESCRIPTION OF THE DRAWINGS

[0006] The present invention is illustrated by way of example and not limitation in the figures of the accompanying drawings, in which like references indicate similar elements and in which:

[0007] Figure 1 illustrates an exemplary network architecture in which an embodiment of the present invention may be implemented;

[0008] Figure 2 illustrates components of a data locking module according to one embodiment of the present invention;

[0009] Figure 3 illustrates components of a backend data store layer according to one embodiment of the present invention;

[0010] Figure 4 illustrates components of a user's data database according to one embodiment of the present invention;

[0011] Figure 5 is a flow diagram showing a process of determining a data locking level within a database according to one embodiment of the present invention;

[0012] Figure 6 illustrates a processing system according to one embodiment of the present invention.

DETAILED DESCRIPTION

[0013] An algorithm for improving database concurrency is described. Note that in this description, references to "one embodiment" or "an embodiment" mean that the feature being referred to is included in at least one embodiment of the present invention. Further, separate references to "one embodiment" in this description do not necessarily refer to the same embodiment; however, neither are such embodiments mutually exclusive, unless so stated and except as will be readily apparent to those skilled in the art. Thus, the present invention can include any variety of combinations and/or integrations of the embodiments described herein.

Exemplary Architecture

[0014] Figure 1 illustrates an architecture in which a method and apparatus of the present invention may be implemented according to one embodiment of the invention. A user may access a server access application 105 that may run on a client machine 100. The server access application 105, e.g. an email client, may provide the user with access to content located on a server 120, which the user may specify via a user interface 110. The server 120 may contain a backend data store 130 comprising data that the user may wish to obtain access to. The backend data store 130 may contain user's data, e.g. email messages, 310 database and deadlock history 315 database illustrated in Figure 3. The server 120 may also contain a data locking module 125. The components of the data locking module are illustrated in Figure 2. In the illustrated embodiment the data

locking module 200 contains a data locator 205, a deadlock analysis module 210 and a hashing module 215. The functions of the data locking module 200 and its components will be described in detail in the following description.

[0015] The physical processing systems which embody the server 120 and the client 100 may include processing systems such as conventional personal computers (PCs) and/or server-class computer systems according to one embodiment of the invention. Figure 6 illustrates an example of such a processing system at a high level. The processing system of Figure 6 may include one or more processors 600, read-only memory (ROM) 610, random access memory (RAM) 620, and a mass storage device 630 coupled to each other on a bus system 640. The bus system 640 may include one or more buses connected to each other through various bridges, controllers and/or adapters, which are well known in the art. For example, the bus system 640 may include a 'system bus', which may be connected through an adapter to one or more expansion busses, such as a peripheral component interconnect (PCI) bus or an extended industry standard architecture (EISA) bus. Also coupled to the bus system 640 may be the mass storage device 630, one or more input/output (I/O) devices 650 and one or more data communication devices 660 to communicate with remote processing systems via one or more communication links 665 and 670, respectively. The I/O devices 550 may include, for example, any one or more of a display device, a keyboard, a pointing device (e.g., mouse, touchpad, trackball), an audio speaker.

[0016] The processor(s) 600 may include one or more conventional general-purpose or special-purpose programmable microprocessors, digital signal processors (DSPs), application specific integrated circuits (ASICs), or programmable logic devices (PLD), or a combination of such devices. The mass storage device 630 may include any one or more devices suitable for storing large volumes of data in a non-volatile manner, such as magnetic disk or tape, magneto-optical storage device, or any of various types of Digital Video Disk (DVD) or Compact Disk (CD) based storage or a combination of such devices.

[0017] The data communication device(s) 660 each may be any devices suitable for enabling the processing system to communicate data with a remote processing system over a data communication link, such as a wireless transceiver or a conventional telephone modem, a wireless modem, an Integrated Services Digital Network (ISDN) adapter, a Digital Subscriber Line (DSL) modem, a cable modem, a satellite transceiver, an Ethernet adapter, or the like.

Methodology

[0018] With these concepts in mind an embodiment of the present invention can be further explored with reference to Figure 5. Figure 5 shows a process of dynamically determining a data locking level. At 510 a user may log-in into an email system via the server access application 105 that may be executed on the client machine 100. The server access application 105 may be, for example, a Microsoft Outlook email server provided by Microsoft Corporation of Redmond, Washington or Eudora email server provided by Qualcomm Inc. of

San Diego, California. The server access application 105 provides the user with the user interface 110 to facilitate a user-friendly access to the server 120, which in one embodiment is a mail server. In one embodiment, the server access application 105 prompts the user via the user interface 110 to enter his/her user name and password in order to log-in into an email system located on the server 120. Upon the user logging into the system, the identification data is transmitted to the server 120. Upon receiving the identification data the server 120 verifies the user's authenticity, and if the identification is confirmed, the user is provided with access to the email system located on the mail server. The techniques for verifying the user's identity are well known in the art and do not require any further explanation.

[0019] At 510 after obtaining access to the email system the user may specify a mailbox that he/she would like to access. In one embodiment the user may have a number of different mailboxes. For example, the user may have an inbox mailbox that may contain the user's new and already read relevant email messages. The user may also have a junk mailbox that may contain spam email that may be filtered according to some predetermined rules. The user may also have a trash mailbox containing messages that the user previously deleted. In one embodiment if the user does not specify which mailbox to access, the user is provided with default mailbox that may be the inbox mailbox.

[0020] At 515 of Figure 5 the data locator 205 of the data locking module 200 illustrated in Figure 2 determines the location of the user-specified data,

which may include a particular mailbox, an email message within a particular mailbox, or a number of email messages within a particular mailbox. In one embodiment every user of the email system is assigned a unique user identifying key. The unique user identifying key may be a unique text string corresponding to a user name, a user password, or combination of both. In one embodiment the unique user identifying key is the user's email address. It will be appreciated that there may be other techniques for developing and assigning a unique user identifying key to every user of the email system.

[0021] In one embodiment of the present invention the backend data store 330 of Figure 3 contains a number of database files located in the user's data 310 database, that may be accessed utilizing unique user identifying keys. In one embodiment hashing may be used to determine a database file corresponding to the user requesting an access to his/her email data. Each database file may contain data corresponding to a number of users according to some predetermined data distribution rules. For example, the data may be distributed between the database files based on a first letter of a user's name. One example of such a data distribution is shown in Figure 4, where a database file 415 contains data of users whose last name starts with letters 'a' through 'k', a database file 420 contains data of users whose last name starts with letters 'l' through 'q', and a database file 425 contains data of users whose last name starts with letters 'r' through 'z'.

[0022] Upon determining a database file which contains data corresponding to the user, the data locator 205 determines a location of the requested data within the file, i.e. data locality. In one embodiment, the data locator 205 after locating the data within the database file that pertains to the user, locates the user-specified mailbox or the user-specified email message within the user-specified mailbox. In one embodiment every mailbox and every email message within every mailbox are assigned a unique identification keys that may be utilized to determine the location of the requested data within a data file. In one embodiment hashing algorithms may be utilized along with the unique identification keys to determine the requested data locality.

[0023] Upon locating the requested data within a data file containing data corresponding to the user, the deadlock analysis module 210 at 530 of Figure 5, in order to ensure data consistency and data concurrency in the database, determines a level of data locking to be applied by accessing a deadlock history database 315. In one embodiment the deadlock history database 315 contains information about previous deadlocks occurred during a predetermined time interval that involved data located within a single database file. The deadlock history database 315 may also contain a number of successful message data requests without a deadlock from the database file during a predetermined time interval. The entries of the deadlock history database 315 corresponds to each data locality. In one embodiment the data locality is defined as a database object within a database file and the deadlock history information is kept for every

database object within a database file. In another embodiment the data locality is defined as a database page and the deadlock history information is kept for every database page within a database file. In yet another embodiment the data locality is defined as a database row and the deadlock history information is kept for every database row within a database file. It will be appreciated that neither the data locality definition nor the deadlock history database design is limited to the examples presented above and may be defined and designed to accommodate the system's needs, data and size in order to reduce data overhead occurring due to maintenance of deadlock information in the database.

[0024] In one embodiment of the present invention, hashing is utilized to determine the locality of the user-requested data within a database file. The hashing module 215 may determine the locality of data by utilizing a hashing algorithm and unique keys assigned to the users of the email system and to the email data. In one embodiment the unique message identification key and mailbox identification key are hashed and utilized as an index into a selected database file. In addition, the unique user identification key may be hashed and used as an index to determine a database file containing data corresponding to the user. Hashing algorithms and hashing techniques are well known in the art and do not require further explanation.

[0025] In one embodiment of the present invention the deadlock analysis module 210 determines the level of data locking based on the deadlock history information stored in the deadlock history database 315. Upon retrieving the

data history information corresponding to the data locality of the user requested data, the deadlock analysis module 210 uses a predetermined deadlock threshold levels to determine the level of locking. In one example if the number of deadlocks for a particular data locality occurred during a predetermined time interval is greater than a number of the allowed deadlocks represented by a high deadlock threshold, the data will be locked more conservatively. If the number of deadlocks for a particular data locality during a predetermined time interval is less than the number of the allowed deadlocks, then a less conservative data locking approach may be used. For example, if the number of the allowed deadlocks is 6 deadlocks in 10 minutes, and the number of deadlocks which actually occurred in the last 10 minutes involving the data locality corresponding to the user-requested data (e.g. an email message in an inbox mailbox) is 7 deadlocks, the inbox mailbox containing the user-requested email message may be locked from access of other users to ensure data consistency. However, if the information retrieved from the deadlock history database 315 indicates that there were no deadlocks that occurred involving the requested email message in the last 10 minutes, only the user-requested email message may be locked from access by other users, thus allowing other users to access the mailbox containing the requested message, which is providing a higher level of data concurrency. The deadlock history database 315 may be updated upon an occurrence of a deadlock in the system.

[0026] In one embodiment of the present invention the data locking levels are database file locking level, database record locking level, database page locking level, database row locking level and database object locking level. It will be appreciated that the data locking levels are not limited to the ones listed above.

[0027] In one embodiment the data locking method is utilized when the user is attempting to perform a writing operation, examples of which may be removing an email message from a mailbox; adding a new mailbox; moving messages from one mailbox to another; changing a parameter, which is utilized to distribute users' data among database files, for example changing user's last name. The determination of the data locking level is performed dynamically upon the user specifying the data to be accessed.

[0028] It will be appreciated that the above described method and apparatus are not limited to email systems and may be utilized with any data that may be represented by a unique identification keys. For example, the above described method and apparatus may be utilized in databases including electronic pager or instant message data, digitally stored video or image information, etc.

[0029] It will also be appreciated that the present invention is not limited to traditional client-server systems and may be implemented in other environments, such as peer-to-peer systems or in non-network environments.

[0030] In addition, it will be recognized that many of the features and techniques described above may be implemented in software. For example, the described operations may be carried out in the server 120 or other suitable device in response to its processor(s) executing sequences of instructions contained in memory of the device. The instructions may be executed from a memory such as TAM 73 and may be loaded from a persistent store, such as a mass storage device, and/or from one or more other remote processing systems. Likewise, hardwired circuitry may be used in place of software, or in combination with software, to implement the features descried herein. Thus, the present invention is not limited to any specific combination of hardware circuitry and software, nor to any particular source of software executed by the processing systems.

[0031] Thus, a method and apparatus for improving database concurrency have been described. Although the present invention has been described with reference to specific exemplary embodiments, it will be evident that various modifications and changes may be made to these embodiments without departing from the broader spirit and scope of the invention as set forth in the claims. Accordingly, the specification and drawings are to be regarded in an illustrative sense rather than a restrictive sense.